<u>Programming 1 Final Project</u>

Name	**Id**
Abdelrahman Amr Ali Mohamed	9010
Abdelrahman Ahmed Hafez	8901
Abdelrahman Hussein Attia	8976
Youssef Mohamed Omar Youssef	8821
Omar Waleed ElSayed Mohamed ElSenousy	8873

1- Introduction

1.1 purpose

This report is being written as a response to the faculty members of the programming 1 course to write a report on our project.

1.2 Background

The project was about developing a bank management system to keep track of the accounts inside the bank. Each account should have an account number, name, mobile, email address, balance, and the date the account was opened. The information about the accounts will be saved in a text file, and there will be another text file to save the employees' usernames and passwords. The program was required to have some functions that will be explained in this report. Those function are login, load, query, advanced search, add, delete, modify, withdraw, deposit, transfer, report, print, save, quit, and menu.

1.3 scope

This report will cover the functions of the program with an explanation of how the function was achieved, some sample runs of the program, the main algorithms used, and a user manual.

2- Functions

2.1 Header Files

Our code is divided into multiple source and header files which allow for better organization and modulation of the code. Our files were as follows:

- struct.h: contains the struct definitions (date, account, user)
- utils.h: contains the common utility functions used across the different functions in our code. Besides including the necessary libraries used throughout the code.
- UI_load.h: contains the user interface and file loading functions like login, menu, load
- options.h: contains the functions called by the menu like add, search, withdraw and print_accounts. The header file only contains the functions meant for external use keeping implementation details within the file itself.

2.2 Utility functions

void format_name(char *name, int len)

- This function takes name and length of input name. The function ensures that only the first letters of the first and last names are capital sized by converting all letters to lowercase and then capitalizing the first letter of the string and the first letter after space character.

int valid_option(int no_of_options)

- This function validates a user input option, ensuring it is a digit, within the specified range of options, and not equal to zero. It relies on validate_digit and returns the validated option as an integer

char validate_digit()

- This function reads user character input from the standard input, ensuring it is a digit and has a length of one character and returns that validated character.

is_duplicate(acc_no[])

- makes sure that entered account number does not already exist in accounts.txt file by iterating over the file using the sscanf() to set a custom delimiter.

enter_valid_acc()

- makes sure the user enters a valid account number by checking for length and non-digit characters.

check_name(name)

- makes sure that the entered name does not contain ‘,’.

check_email(email)

- makes sure that entered email contains ‘@’ and ‘.com’ and does not contain ‘ ‘.

check_bal(temp)

- makes sure balance is greater than zero and is all digits.

check_mobile(mobile)

- makes sure that the entered mobile number is all digits and is of appropriate length.

int no_accounts(FILE *f)

- counts the number of entries in a file.

const char *month_name(int n)

- This function takes the month number and returns the month name.

2.3 The main

The main.c file will start with the start of the program. In this file, the global pointer to the struct account is defined and initialized with null and the global integer variable count_acc is defined and initialized with zero. The main function then displays a welcome message and calls the call menu function.

2.4 Menus

2.4.1 call_menu

- The call_menu() function's purpose is to iterate calling the main menu or the starting menu - which will be explained next -while reducing stack until the user choose to quit.
- It operates in infinite loop with a condition that if a local integer variable called (x) is equal to one, it will call the starting menu, then after return it will call the main menu.
- The variable (x) is initiated with one before the loop to guarantee calling the starting menu in the first iteration.
- It has another condition that if count_acc is equal to zero, it will call the empty file menu.

2.4.2 starting menu

- The starting menu is a function that has been called startup().
- It gives the user two options: to login or to quit
- The user should input 1 to login and 2 to quit.
- If the user input something other than 1 or 2, the user will be prompted for input again until he inputs a valid input.
- If the user inputs 1, the login function will be called, and if he inputs 2, the quit function will be called.
- After the program returns from the login function, it will return to the call menu.

2.4.3 empty file menu

- The function is called empty_file_menu()
- It is only called if the file containing the information of the accounts, called "accounts.txt", is empty or if it doesn't exist and the user want to create it.
- It opens the file in write mode to create it if it doesn't exist, then the file is closed.
- It only gives the options of add and quit as no accounts are saved to perform other functions.
- It has the same validation for the input as the starting menu.

2.4.4 Main menu

- The function is called menu()
- It gives the user to choose from 11 options: add, delete, modify, search, advanced search, withdraw, deposit, transfer, report, print, quit
- Inputting the number of options will call the corresponding function and after returning from that function, it will return zero to the call menu. Otherwise, if the user inputs something other than the 11 options, the function will return 1 to the call menu.

2.5 Login

- The function is called login()
- Two variables of the struct type user are declared named entered and saved, where entered will save the data inputted by the user, while saved will save the data read from the file named "users.txt".
- After the user inputs his username and password, it will be checked with every username and password in the file. If there is a match, the file will be closed, any allocated will be freed, and it will check if the accounts array is null, where if it is, the load function will be called.
- If there is no match found, the file pointer is set to the begging of the file, and the user is asked to input his username and password again.
- The user only has five attempts to enter a correct username and password. After that the program will terminate with an error message after closing the file and freeing any allocated memory.

2.6 Load

- The load function is called load()
- The first thing the function does is that it opens the "file accounts.txt" in read mode.
- If the pointer to the file is equal to null, it will display to the user the file wasn't found and give him two options of quitting and putting the file in the right place before reopening the program, or creating the file from scratch
- If the user chooses the first option, the function quit will be called with a value of 1 indicating an error relating to files.
- If the user chooses the second option, the function will return to login.
- The second thing it does is checking if the file is empty by getting the first character in it. If it is equal EOF, the function will return to login.
- Then, the function no_accounts() is called and the value returned from it is assigned to count_acc.
- After that, Memory is allocated to the global pointer accounts to turn it into an array of structs. The memory allocated to it is equal to count_acc multiplied by the size of the account type.

- Then, we check if the memory was actually allocated or if accounts is still null. If it is still null, the program will be terminated with a value of 2, indicating a problem with memory after displaying a message informing the user that there is not enough space to load the data.
- We then use fseek() to make sure the file pointer is moved to the beginning of the file.
- After that, the character arrays buffer, name_temp, email_temp, and date_temp are declared. These strings will be used to in filling the array.
- The next step is to fill the array with the accounts' data saved in the file. This is done using a for loop iterating count_acc times with an index starting from zero which is incremented in each iteration.
- Then, a line is read from the file using fgets() and put in buffer. If the return from fgets() is null, the program will be terminated with a value of 1 indicating an error with files after displaying a message saying than there was an error while reading from the file.
- The data in buffer is separated by a comma as the delimiter. sscanf() will be used to put the account number, balance, and mobile number in their members in the first element of the array while the name, email and date will be saved in name_temp, email_temp, and date_temp respectively.
- It is then checked if date_temp has a newline character at the end which will be removed if found.
- sscanf() is then used to save the date in its member in the first element of the array.
- Memory is then allocated to the name and email members of the first element of the array with length of the string saved in name_temp and email_temp plus one.
- The strings in name_temp and email_temp are then copied to their members of the first element of the array.
- These steps are then repeated for the rest of the elements in the array until index is equal to count_acc.
- The file is then closed and the function will return to login.

2.7 Add

- void add()
- A new struct of type account is created to be filled by the user.
- The user is prompted to enter a new account number, validates it, and makes sure it doesn't already exist in the accounts.txt file using the is_duplicate() function. If the account number already exists, an error message is displayed, and the function returns to the main menu.
- The user is prompted to enter a new name, email, balance, and mobile number, which are all validated for errors and shortened to their actual length by replacing the new line

character ‘\n’ with a null character ‘\0’. If the input is invalid, an error message is displayed, and the user is prompted to enter the value again.
- The system’s current month and year are taken using the time.h library and saved to the struct.
- The user is then prompted to choose to either save or not save.
- Choosing save creates a new report file corresponding to the account number using ssprintf to point to the directory and fopen to write, increases the number of accounts global variable (count_acc) by one, resizes the global array of structs to its appropriate size using realloc, adds the newly created struct to the global array of structs, and saves to overwrite the accounts.txt file.
- Choosing not to save returns to the main menu without changing anything.
- Allocated memory is freed in either save options.

2.8 Delete

- Void DELETE()
- The user is prompted to enter a valid account number that is validated and checked to make sure it exists by iterating over the existing global array of structs. If it doesn’t exist an error message is displayed.
- When an account number is entered, its balance is checked to make sure its balance is zero. If the account’s balance is greater than zero, an error message is displayed as an account with a balance more than zero cannot be deleted.
- The user is prompted to either save, which removes the account by equaling it with the last account in the global array then removing the last account by decrementing the number of accounts (count_acc) by one then using realloc with the new count_acc.
- The report file corresponding to the account number is removed.
- The function then returns to the main menu.

2.9 Modify

- void modify()
- The user if prompted to enter a valid account number that is validated and checked to make sure it exists by iterating over the global array of structs and comparing the entered account number with the account number of each struct in the array. An error message is displayed if the account number is invalid or doesn’t exist.
- The user is prompted to choose a value to modify (name, email, or mobile) or “done” which saves by overwriting the existing accounts.txt file.
- When changing a value, temporary buffer is allocated and replaces the previous value using stcpy(,) then freed.
- Each value the user inputs is validated and checked for errors.
- When the user chooses to modify a certain account detail, the pervious value of such account detail is displayed to the user.

- After a value is modified, the user is prompted to either save, which edits the account struct corresponding to the entered account number, or not save which does nothing and returns back to the modify choices menu.
- When the choice “done” is chosen, changes are saved by overwriting the existing accounts.txt file, and the user is returned to the main menu.

2.10 Search

- The function is called when the user press on the search option in the menu
- The function takes no parameters
- It asks the user to input the account number he is searching for
- Validation is done on the account number entered by the user to check that consist of 10 characters and these 10 characters are numbers
- If the account number form is corrected, I search for this account number in the all account numbers the bank system have.
- I made a loop that keep searching in the account numbers we have until we find the details of the account that user want his details. If found I exit the for loop and change the flag value indicating that the account value is found and I print all the details of the account with this account number also I call the month name function since the data is stored as digits 12-2004 so I send the month number and it returns the month name so I can print the data in that format December 2004.l.
- If the account number is not found in the account numbers we have then the value of the flag is not updated it indicates I didn’t enter the conditional statement in my code and update my flag value.
- At the end I do a conditional statement to see if the flag not updated than will I print a statement indicating that the account with this account number doesn’t exist in the bank management system.

2.11 Advanced search

- The user enters the keyword which is part of a name
- I compare it with every name of customers in the database of bank management system using a loop. If found a customer name that contain that keyword, I print all his account details then I keep searching until I print data of all customers that has the keyword entered by the user in their names. Also, I convert the date which is stored in the bank system for each customer in this form 12-2004 using a function I call, sending to it the month name which is 12 and I return is as a word(string) December and print it.
- I made a counter and initialized it to 0 if it’s not updated or incremented within the for loop it indicates that no accounts found containing that key word in their names and I output to user that no accounts containing this keyword in their name were found.

2.12 Save

- After each operation like add, modify, withdraw, transfer, deposit, delete the user is asked whether he would like to save or no if yes then he is directed to the save function. I open the accounts.txt (the file where data of all customers is stored) and overwrite it using the new data that has been modified for that account owner.

2.13 Withdraw

void withdraw()

Input: Account number

How it works:

- First, the function asks the user to input his account number in a do while loop.
- Validation function returns 1 if one of the tests failed other than that it returns 0. So, the prompt is in a do while loop that asks the user for the account number every time one of the tests fails.
- If the account number exists the function will print to the user that the maximum withdrew money for one transaction is 10000 and ask the user to enter the amount of money to be withdrew.
- If all is good, we take the withdrew money as a string to have the ability to check if he prompted anything else than a digit then turn it to a float number (if it is accepted) by strtof function.
- If he has sufficient money the function asks the user if he wants to save the transaction. If he input yes, the function will remove the withdrew money from the struct and call the save function and print to the user that the transaction was successful. If he says no the function will end without changing anything.

Errors:

[1] Function called validation that check the account number and make sure it is a 10-digit number, and all is digits and make sure there is no negative numbers.

[2] For loop searches if the account number exist or not, if it doesn't it tells the user that the account number doesn't exist and ask him if he wants to exit the function or put another account number (so he doesn't get stuck in the withdraw function).

[3] Do while loop that asks the user for the withdrew money and make sure that the inputted prompt is a number greater than 0 and smaller than 10000 and checks if all are digits.

[4] If the user withdrew more than what he has the function will end printing that there isn't sufficient money.

2.14 Deposit

void deposit()

Input: Account number

How it works:

- First, the function asks the user to input his account number in a do while loop.
- Validation function returns 1 if one of the tests failed other than that it returns 0. So, the prompt is in a do while loop that asks the user for the account number every time one of the tests fails.
- If the account number exists the function will print to the user that the maximum deposited money for one transaction is 10000 and ask the user to enter the amount of money to be deposited.
- If all is good, we take the deposited money as a string to have the ability to check if he prompted anything else than a digit then turn it to a float number (if it is accepted) by strtof function.
- Then, the function asks the user if he wants to save the transaction. If he input yes, the function will add the deposited money in the struct and call the save function and print to the user that the transaction was successful. If he says no the function will end without changing anything.

Errors:

[1] Function called validation that check the account number and make sure it is a 10-digit number, and all is digits and make sure there is no negative numbers.

[2] For loop searches if the account number exist or not, if it doesn't it tells the user that the account number doesn't exist and ask him if he wants to exit the function or put another account number (so he doesn't get stuck in the deposit function).

[3] Do while loop that asks the user for the deposited money and make sure that the inputted prompt is a number greater than 0 and smaller than 10000 and checks if all are digits.

2.15 Transfer

void transfer()

Input: Account number

How it works:

- First, the function asks the user to input the account number of the sender in a do while loop.
- Validation function returns 1 if one of the tests failed other than that it returns 0. So, the prompt is in a do while loop that asks the user for the account number every time one of the tests fails, the same happens with the account number of the receiver.
- If the account number of sender and receiver exist, the function will ask the user to enter the amount of money to be transferred.
- If all is good, we take the transferred money as a string to have the ability to check if he prompted anything else than a digit then turn it to a float number (if it is accepted) by strtof function.

- If the sender has sufficient money the function asks the user if he wants to save the transaction.
- If he input yes, the function will add the transferred money in the struct of the sender and remove the transferred money from the struct of the receiver and call the save function and print to the user that the transaction was successful.
- If he says no the function will end without changing anything.

Errors:

[1] Function called validation that check the account number of sender and receiver and make sure it is a 10-digit number, and all is digits and make sure there is no negative numbers.

[2] For loop searches if the account number of sender and receiver exist or not, if it doesn't it tells the user that the account number doesn't exist and ask him if he wants to exit the function or put another account number (so he doesn't get stuck in the transfer function).

[3] Do while loop that asks the user for the transferred money and make sure that the inputted prompt is a number greater than 0 and checks if all are digits.

[4] If the user (sender) transfer more than what he has the function will end printing that there isn't sufficient money.

2.16 Report And Print

void report()

This function prints the 5 latest transactions done by the user. It is void and has no parameters. It takes account number from user and validates the number to be both a valid account and be of an existing account. Then the transactions are loaded and printed for the user.

Printing and sorting all accounts

void print_accounts()

This function prints the details of user accounts in a structured format. It first sorts the accounts based on user's preference. Then, it iterates through the sorted accounts and prints their details.

void check_for_sort()

This function prompts the user to choose a type of sorting (by name, date, or balance), validates the input, and calls the sort function with the chosen sorting criteria.

void sort(int type)

This function performs a bubble sort on the array of accounts, according to the entered number. It compares between the accounts using compare_vals function which returns 1 if a swap needs to occur. Swap is handled by swap_acc function. This function works on the different sorting preferences because compare_vals handles the different comparison methods.

void swap_acc(account *a, account *b)

This function takes two pointers to account structures as parameters, creates a temporary account structure, and swaps them.

int compare_vals(int j, int type)

compare_vals is responsible for handling the comparisons of different types. In case of name, strcmp is used. In case of date, datecmp is used. And in case of balance, If next is greater than current then it returns 1. A return of 1 indicates a swap.

int datecmp(date a, date b)

This function compares two date structures and returns an integer indicating their relationship. It returns 1 if 1st is older than 2nd, 0 if they are equal, and -1 if 2nd is older than 1st.

2.17 Adding Transactions functions

void add_transaction_to_file(char *acc_no, int type, float amount, char *transfered_acc)

Function called after a transaction(withdraw/deposit/transfer) is made.

This function adds a new transaction entry to the transaction file associated with a given account by using several functions and it:

- Opens the file in read/write mode.
- loads existing transactions into a transaction array.
- Creates the new transaction and adds the entry to the array.
- If the transaction count exceeds a specified limit, the oldest entry is eliminated.
- writes the updated transactions back to the file.

Parameters:

- acc_no: account number for which the transaction file needs to be opened.
- type: number representing transaction type (withdraw/deposit/transfer)
- amount: balance change in the transaction
- transfered_acc : account number for which the transaction was transferred to/from.

NULL if transaction was withdraw or deposit.

FILE *open_tran_file(char *acc_no, char type[3])

This function takes the account number and the file access type as parameters, constructs the file path based on the account number, and opens the corresponding transaction file. It returns a pointer to the opened file.

char* load_transaction(FILE *transactions_file, int no_of_entries, int elimenate)

This function takes the file pointer, the number of transactions, and a flag to eliminate the oldest entry if the entry count exceeds 5 so that it doesn't eliminate in case of print. The transactions are read line by line and stored in a string array.

char* new_transaction(int type, float amount, char *transfered_acc)

This function creates a new transaction entry string based on the transaction type, amount, and, in the case of transfers, the target account. It returns a pointer to the string.

2.18 Quit

void quit(int x)

This function is called upon program termination. It takes x as a parameter which corresponds to an error message or successful termination. Function also frees the loaded array of accounts and the allocated blocks for account names and emails.

3- Algorithms

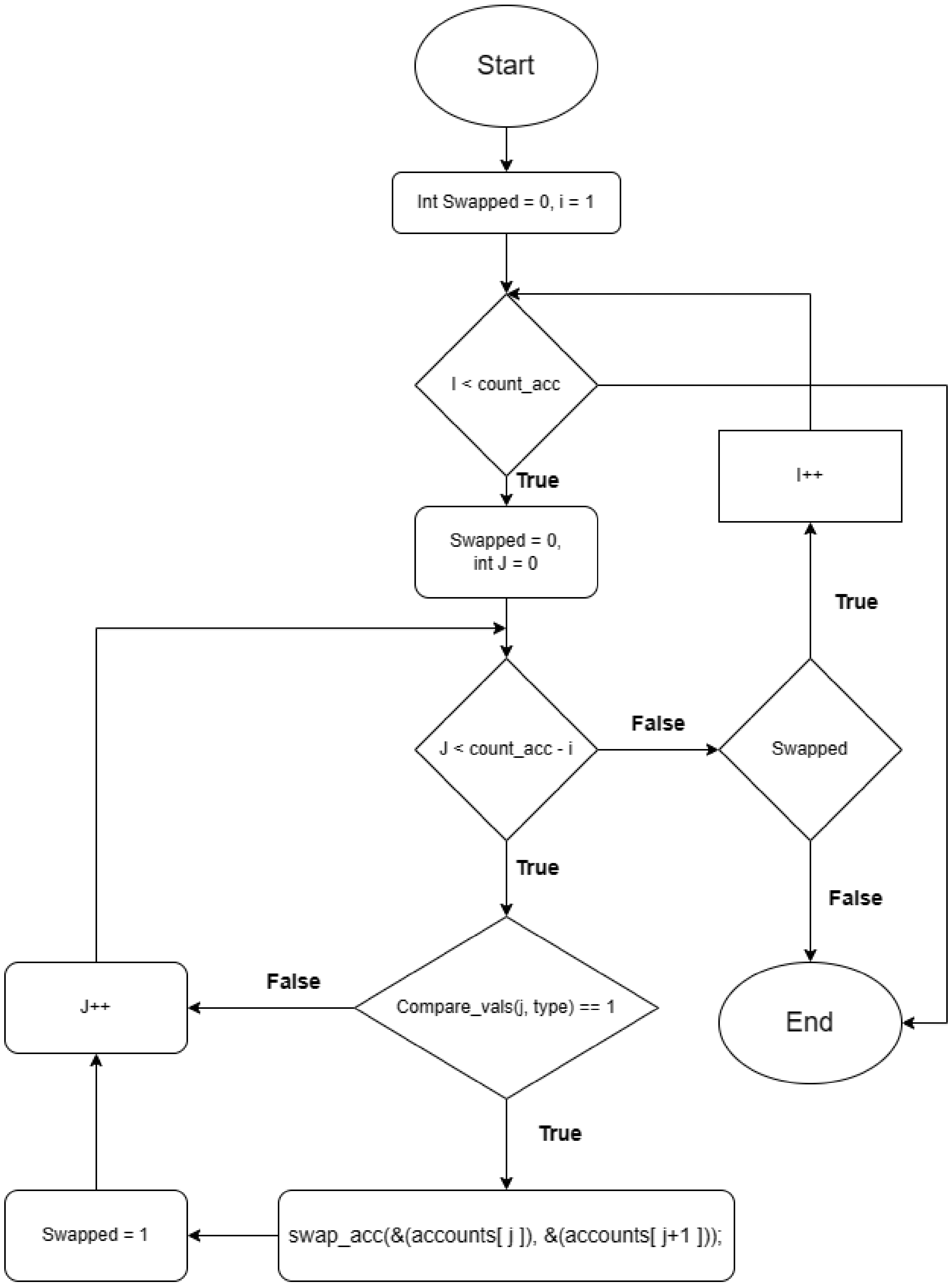
3.1 Searching algorithm (linear search)

- The user enters the account number
- I validate the account number to ensure its consist of 10 digits
- I make a flag and initialize it to 0
- Then I do a loop to pass on our array of structs
- I compare each time the account number entered by the user with an account number in my array of structs by using the strcmp function it will return 0 if accounts match.
- If I found it I print all the details of that account
- I break from the loop and update the value of the flag
- I check the value of flag to see if it not updated than I print no account is found for this account number.

For the advanced search, I would rather look using the key part of the name rather than the account number and if I found someone I print his details until I access all members of the array and here I made a counter and initialized it to 0 if not updated within the loop it indicates no accounts were found containing that key word hence I do a conditional statement to see if the counter is 0 I would print that no accounts found containing that key word in their names.

3.2 Sorting Algorithm

Bubble sort sorting algorithm was used to sort our data.



4- User Manual

Bank Management System

Introduction about the Program

Using this program, you will be access of all costumers of the bank and be able to do several options.

When you start, you will get a welcome message. Then, you will get a menu with these 2 available options: login and quit.

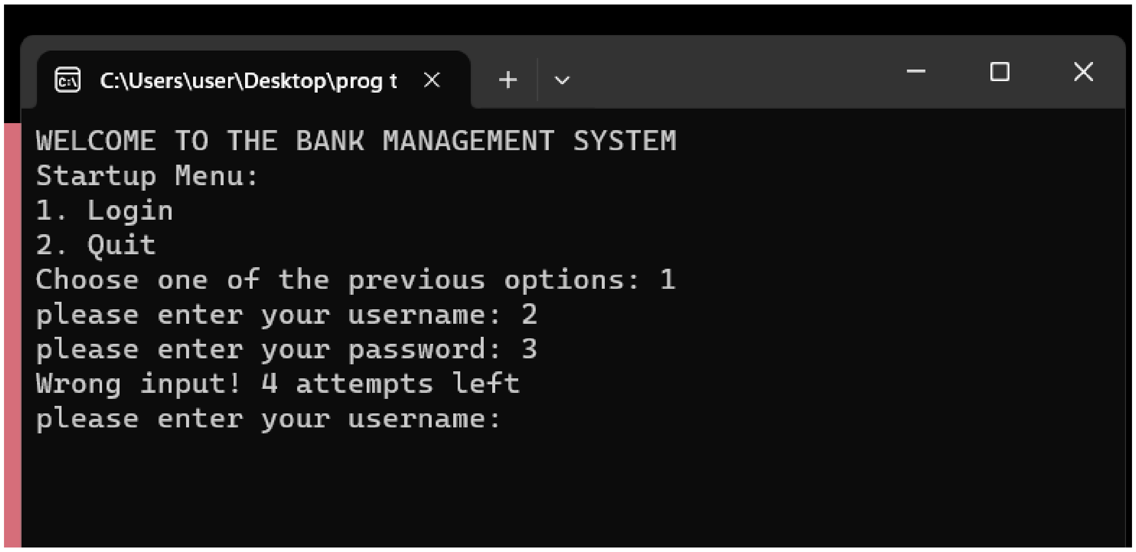
Starting Menu

1-Login

To login press 1

You are required to enter your username and password.

If you enter them wrong, you will have 4 other attempts. If all attempts were wrong you can no longer login into the system and would have to restart the program.



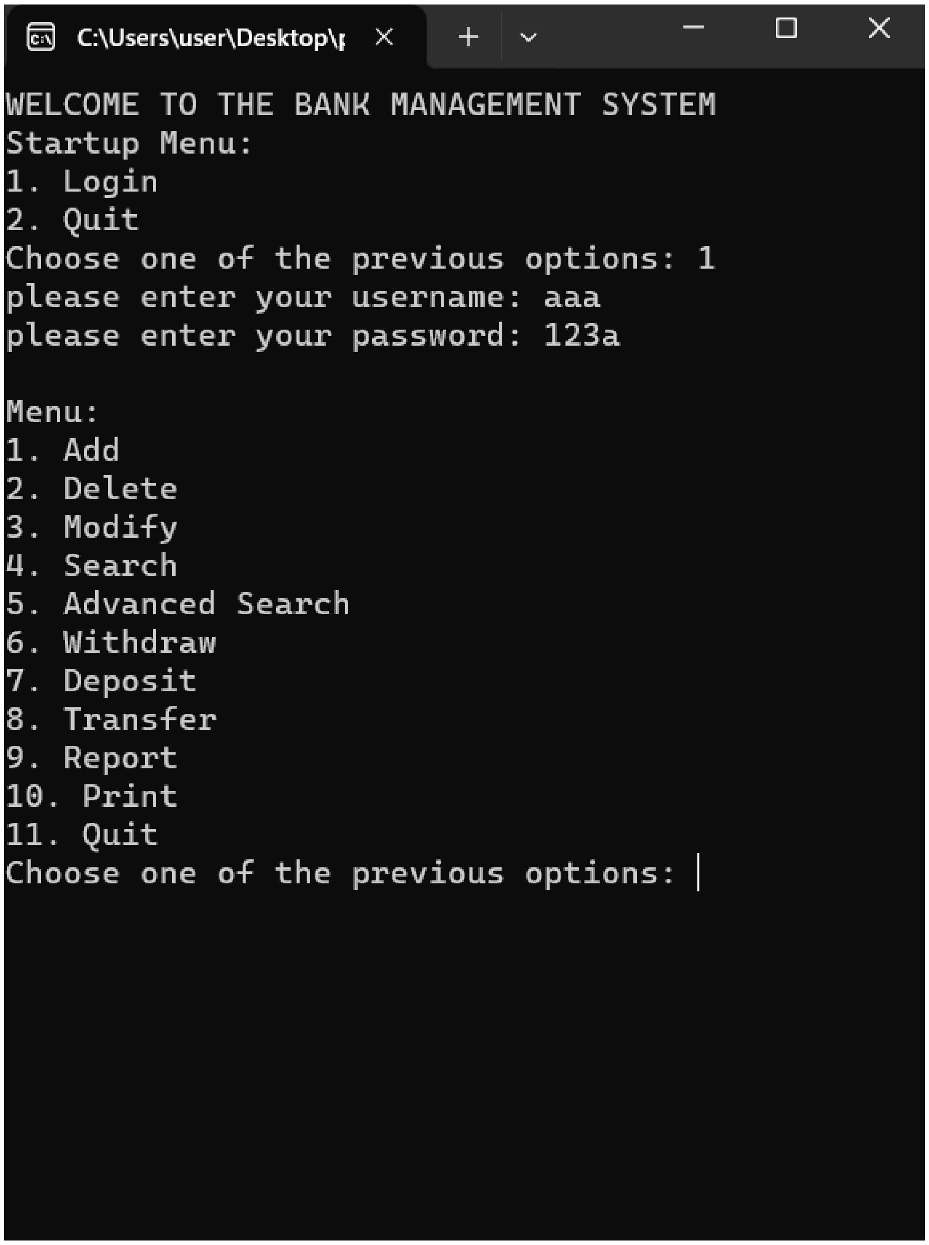
```
WELCOME TO THE BANK MANAGEMENT SYSTEM
Startup Menu:
1. Login
2. Quit
Choose one of the previous options: 1
please enter your username: 2
please enter your password: 3
Wrong input! 4 attempts left
please enter your username:
```

2-Quit

For quit press 2, if you press quit you will close the program.

If you successfully logged in, you will get a menu with all options available

Main Menu



```
WELCOME TO THE BANK MANAGEMENT SYSTEM
Startup Menu:
1. Login
2. Quit
Choose one of the previous options: 1
please enter your username: aaa
please enter your password: 123a

Menu:
1. Add
2. Delete
3. Modify
4. Search
5. Advanced Search
6. Withdraw
7. Deposit
8. Transfer
9. Report
10. Print
11. Quit
Choose one of the previous options: 
```

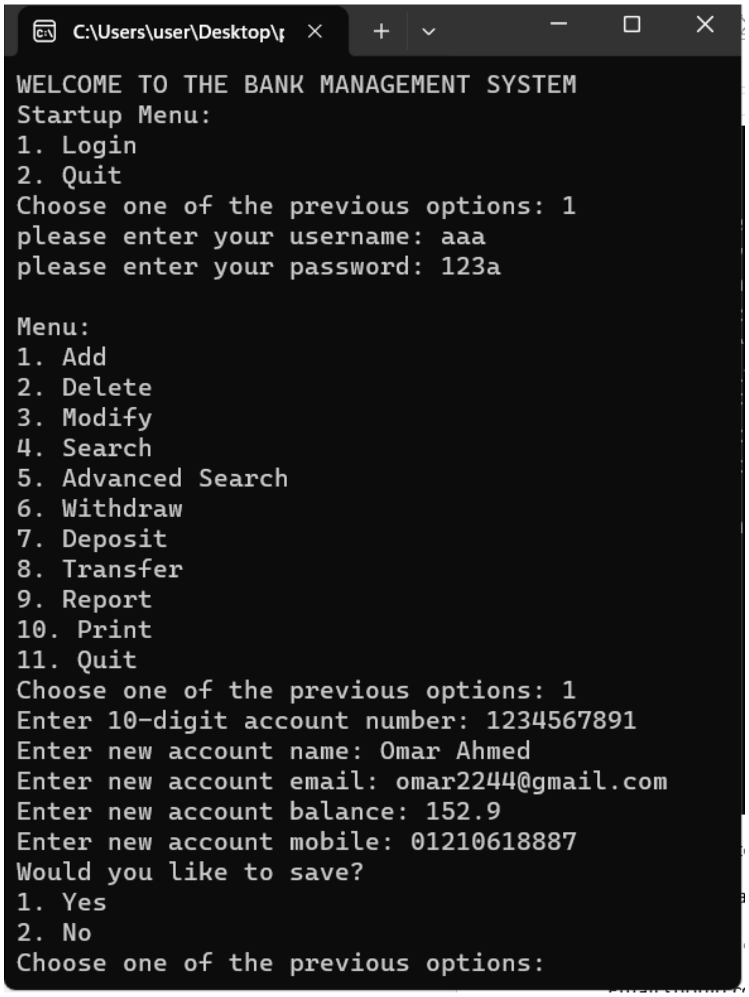
1-Add

Press 1 for adding a new account, you will need to fill all this information

Ensuring that account number is 10 digits

Name doesn’t contain a comma, email should include @ and .com and doesn’t start with @.

At the end you will be asked if you want to save the data.

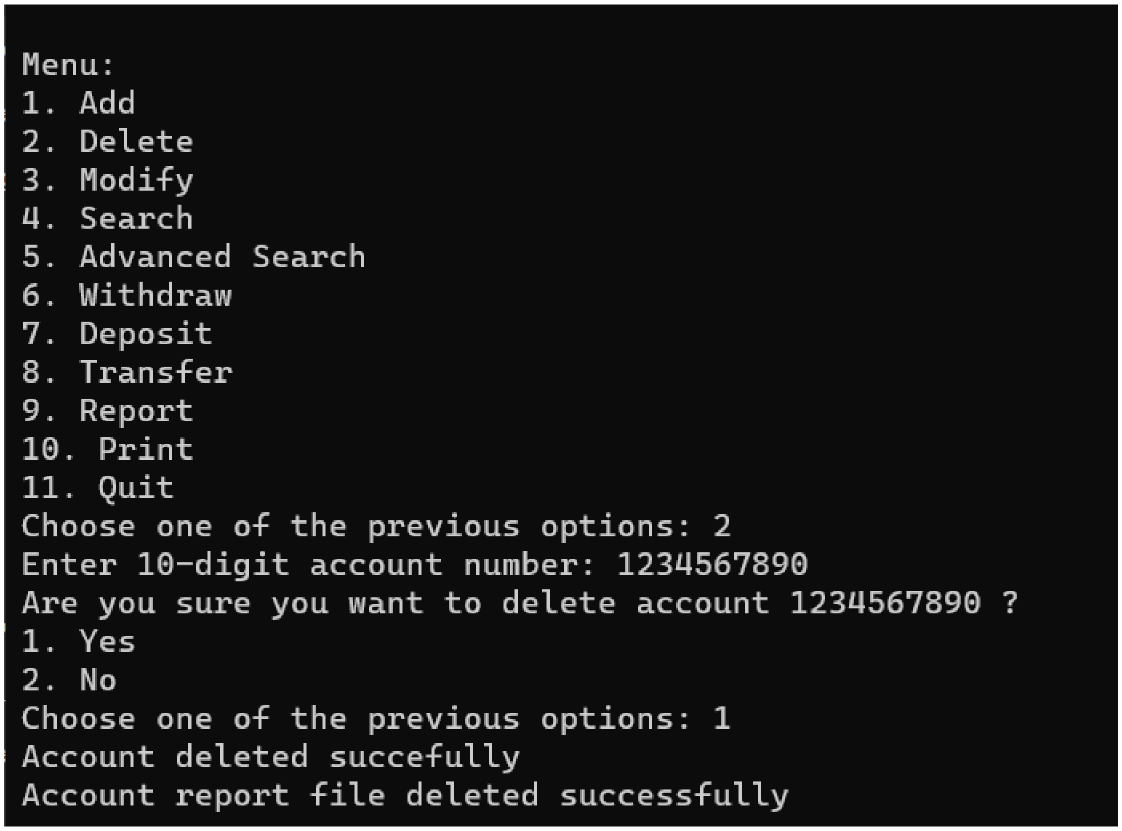


```
WELCOME TO THE BANK MANAGEMENT SYSTEM
Startup Menu:
1. Login
2. Quit
Choose one of the previous options: 1
please enter your username: aaa
please enter your password: 123a

Menu:
1. Add
2. Delete
3. Modify
4. Search
5. Advanced Search
6. Withdraw
7. Deposit
8. Transfer
9. Report
10. Print
11. Quit
Choose one of the previous options: 1
Enter 10-digit account number: 1234567891
Enter new account name: Omar Ahmed
Enter new account email: omar2244@gmail.com
Enter new account balance: 152.9
Enter new account mobile: 01210618887
Would you like to save?
1. Yes
2. No
Choose one of the previous options:
```

2-delete

Press 2 for delete, If the user want to delete the account he should ensure account balance not greater than 0.

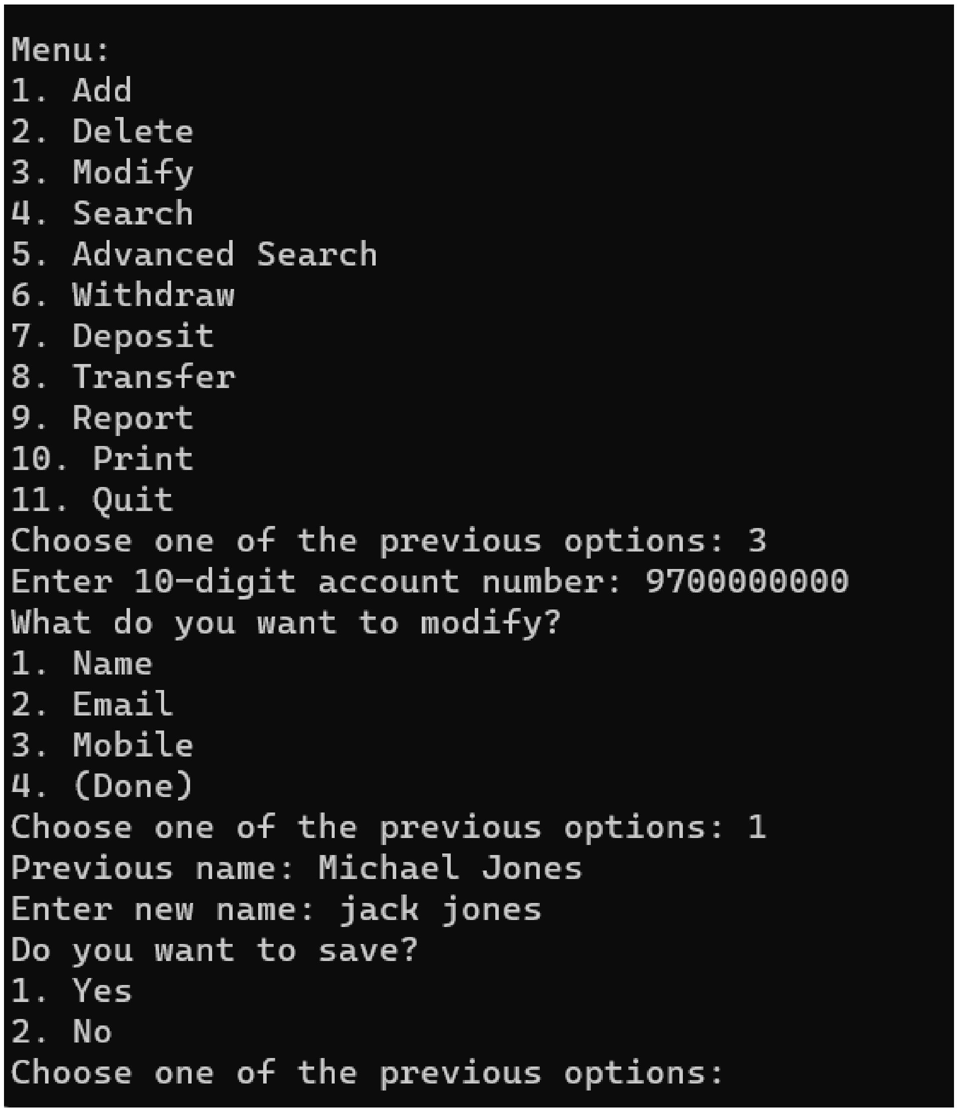
At the end he would be asked if he want to save or no.



```
Menu:
1. Add
2. Delete
3. Modify
4. Search
5. Advanced Search
6. Withdraw
7. Deposit
8. Transfer
9. Report
10. Print
11. Quit
Choose one of the previous options: 2
Enter 10-digit account number: 1234567890
Are you sure you want to delete account 1234567890 ?
1. Yes
2. No
Choose one of the previous options: 1
Account deleted succefully
Account report file deleted successfully
```

3-Modify

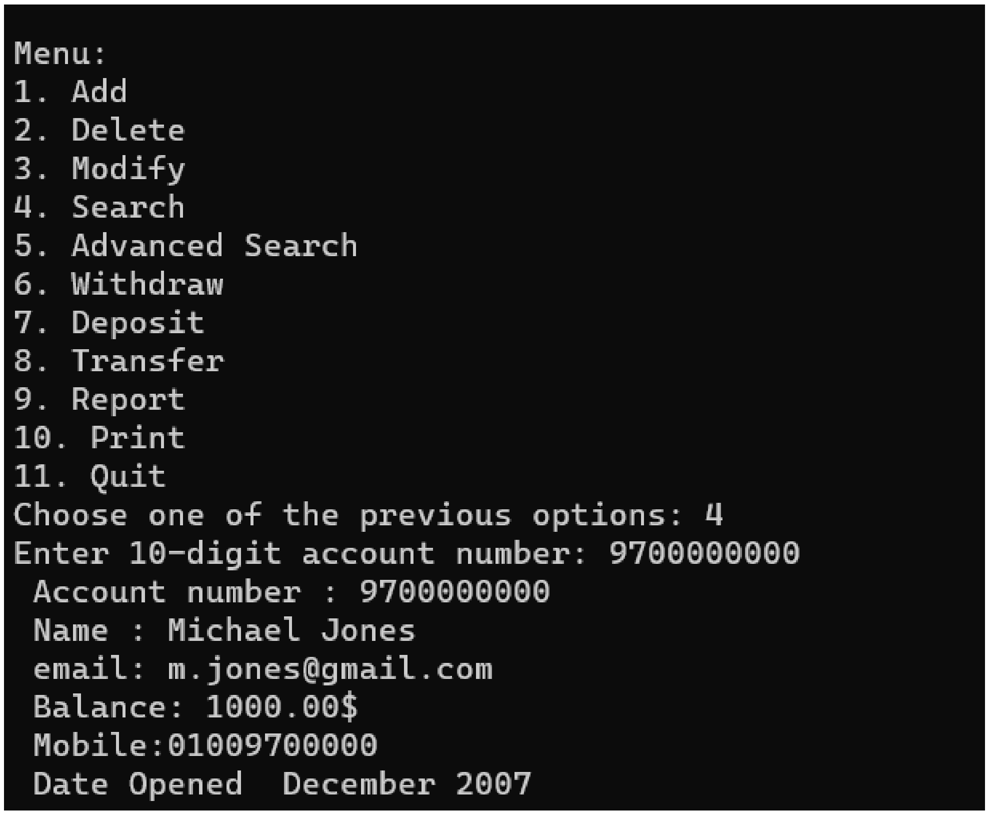
The user should press 3 to modify then choose what he would like to modify from name, email or mobile and at the end he will be asked to save or no.



```
Menu:
1. Add
2. Delete
3. Modify
4. Search
5. Advanced Search
6. Withdraw
7. Deposit
8. Transfer
9. Report
10. Print
11. Quit
Choose one of the previous options: 3
Enter 10-digit account number: 9700000000
What do you want to modify?
1. Name
2. Email
3. Mobile
4. (Done)
Choose one of the previous options: 1
Previous name: Michael Jones
Enter new name: jack jones
Do you want to save?
1. Yes
2. No
Choose one of the previous options:
```

4-search

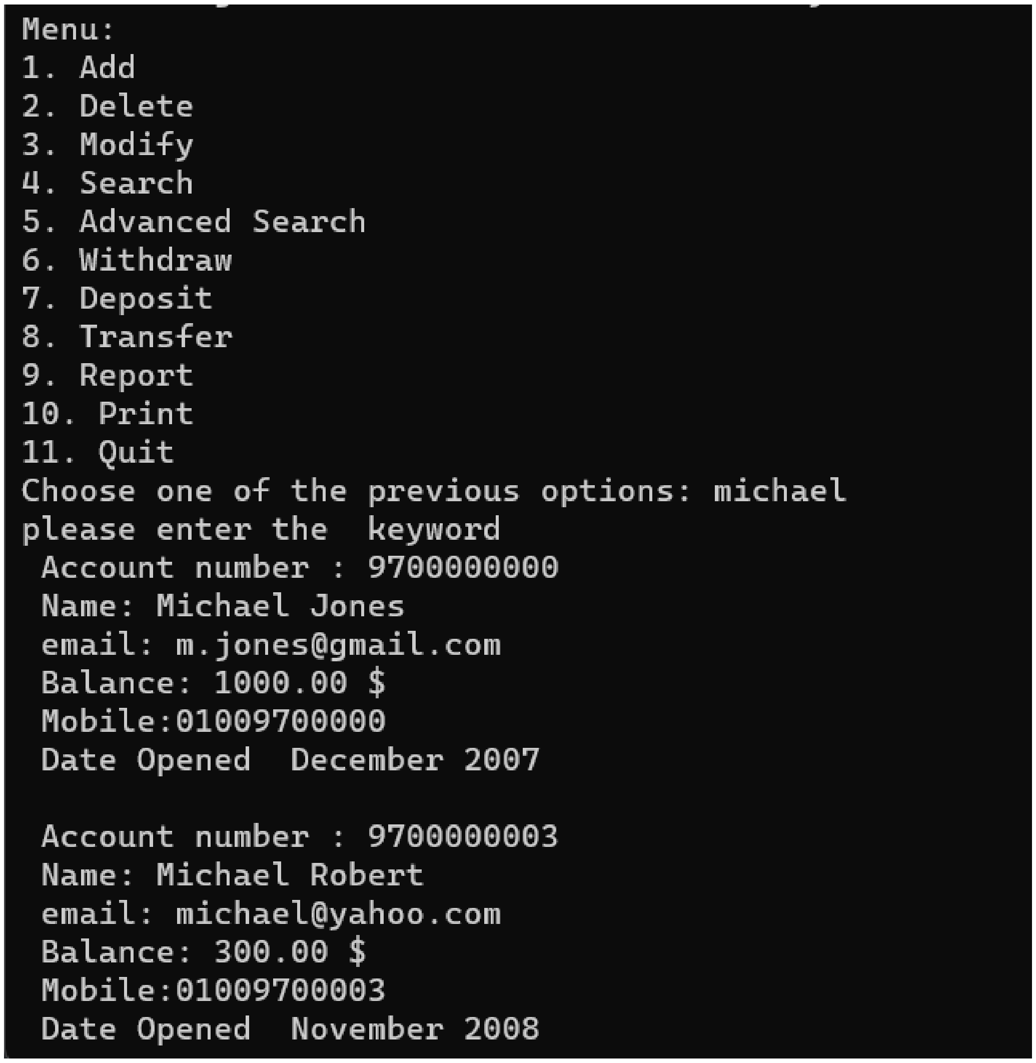
Press 4 for search, you should enter the account number and if found it will print the details of the account



```
Menu:
1. Add
2. Delete
3. Modify
4. Search
5. Advanced Search
6. Withdraw
7. Deposit
8. Transfer
9. Report
10. Print
11. Quit
Choose one of the previous options: 4
Enter 10-digit account number: 9700000000
 Account number : 9700000000
 Name : Michael Jones
 email: m.jones@gmail.com
 Balance: 1000.00$
 Mobile:01009700000
 Date Opened  December 2007
```

5-Advanced Search

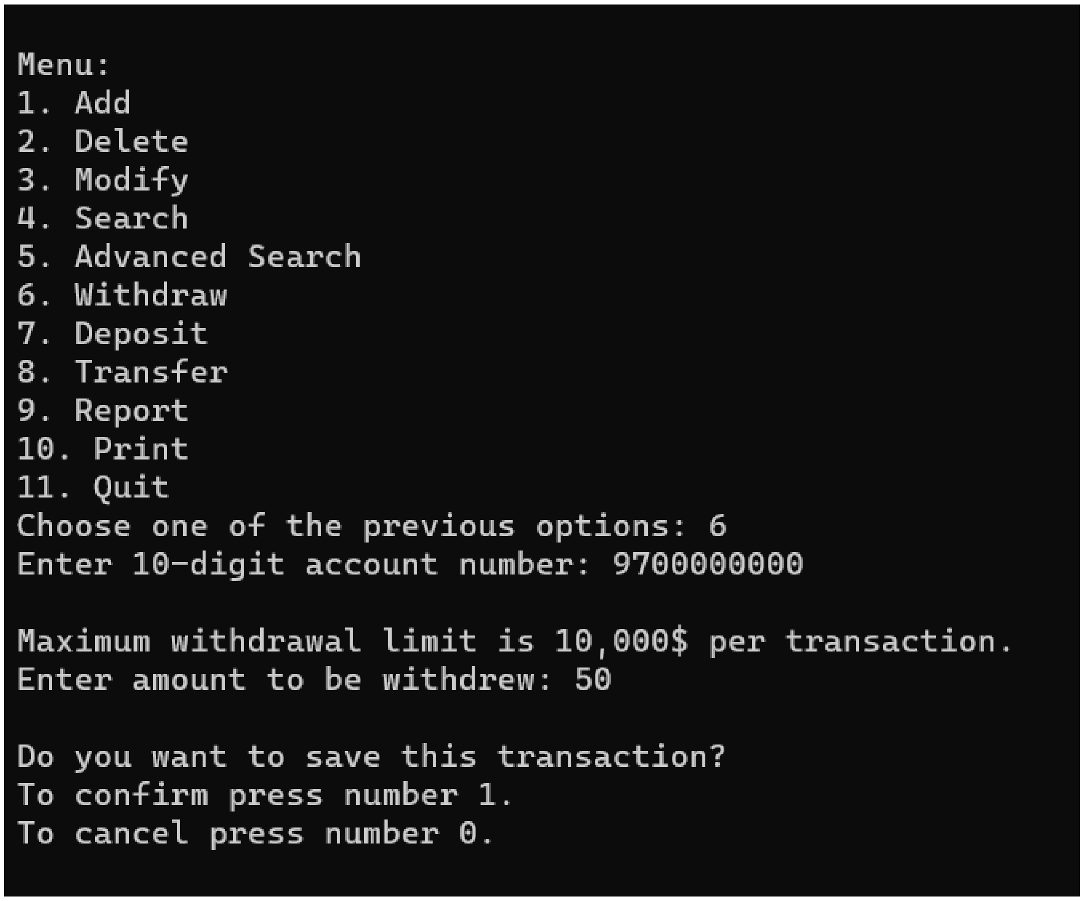
Press 5 for advanced search. You should enter the key word and it will print all accounts with that key word in their names.



```
Menu:
1. Add
2. Delete
3. Modify
4. Search
5. Advanced Search
6. Withdraw
7. Deposit
8. Transfer
9. Report
10. Print
11. Quit
Choose one of the previous options: michael
please enter the  keyword
 Account number : 9700000000
 Name: Michael Jones
 email: m.jones@gmail.com
 Balance: 1000.00 $
 Mobile:01009700000
 Date Opened  December 2007

 Account number : 9700000003
 Name: Michael Robert
 email: michael@yahoo.com
 Balance: 300.00 $
 Mobile:01009700003
 Date Opened  November 2008
```

6-withdraw

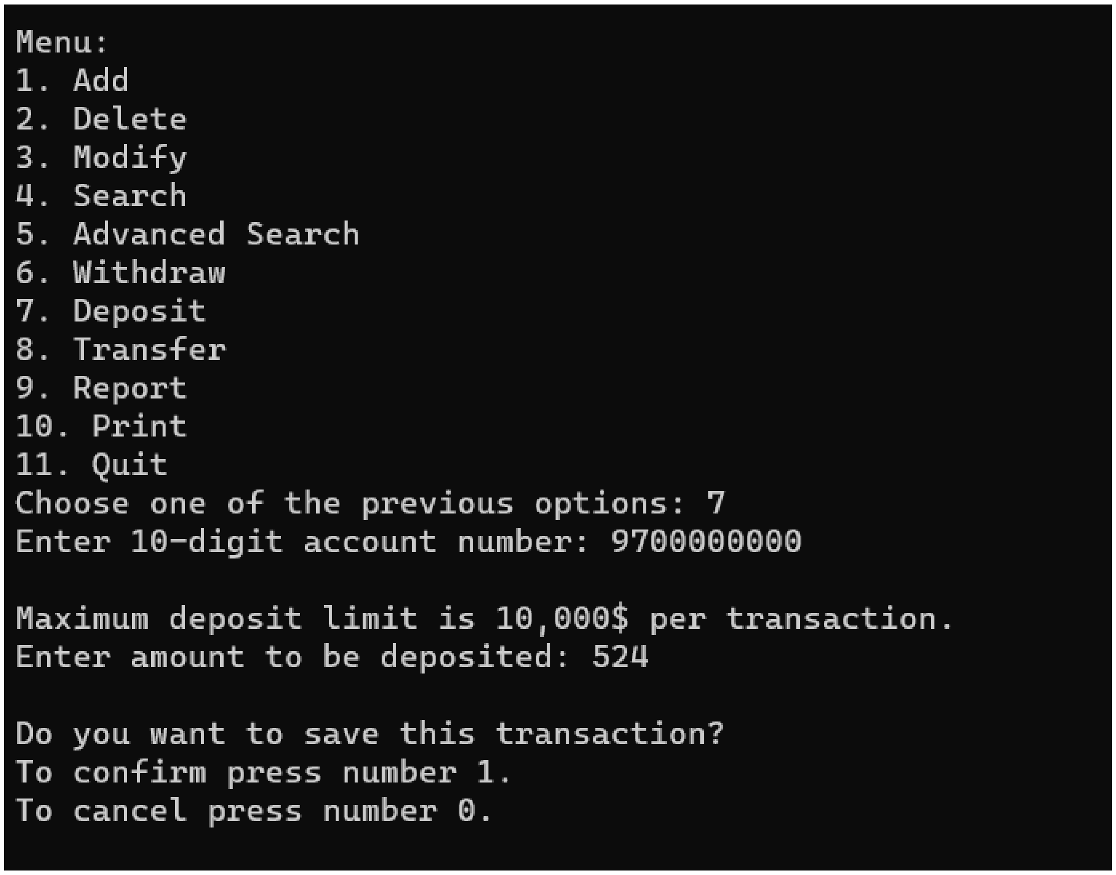
For withdraw press 6, you should enter the account number and amount to be withdrawed ensuring that it is less than 100000.



```
Menu:
1. Add
2. Delete
3. Modify
4. Search
5. Advanced Search
6. Withdraw
7. Deposit
8. Transfer
9. Report
10. Print
11. Quit
Choose one of the previous options: 6
Enter 10-digit account number: 9700000000

Maximum withdrawal limit is 10,000$ per transaction.
Enter amount to be withdrew: 50

Do you want to save this transaction?
To confirm press number 1.
To cancel press number 0.
```

7- Deposit

For deposit press 7, you should enter the account number and amount to be deposited ensuring that it is less than 10000.

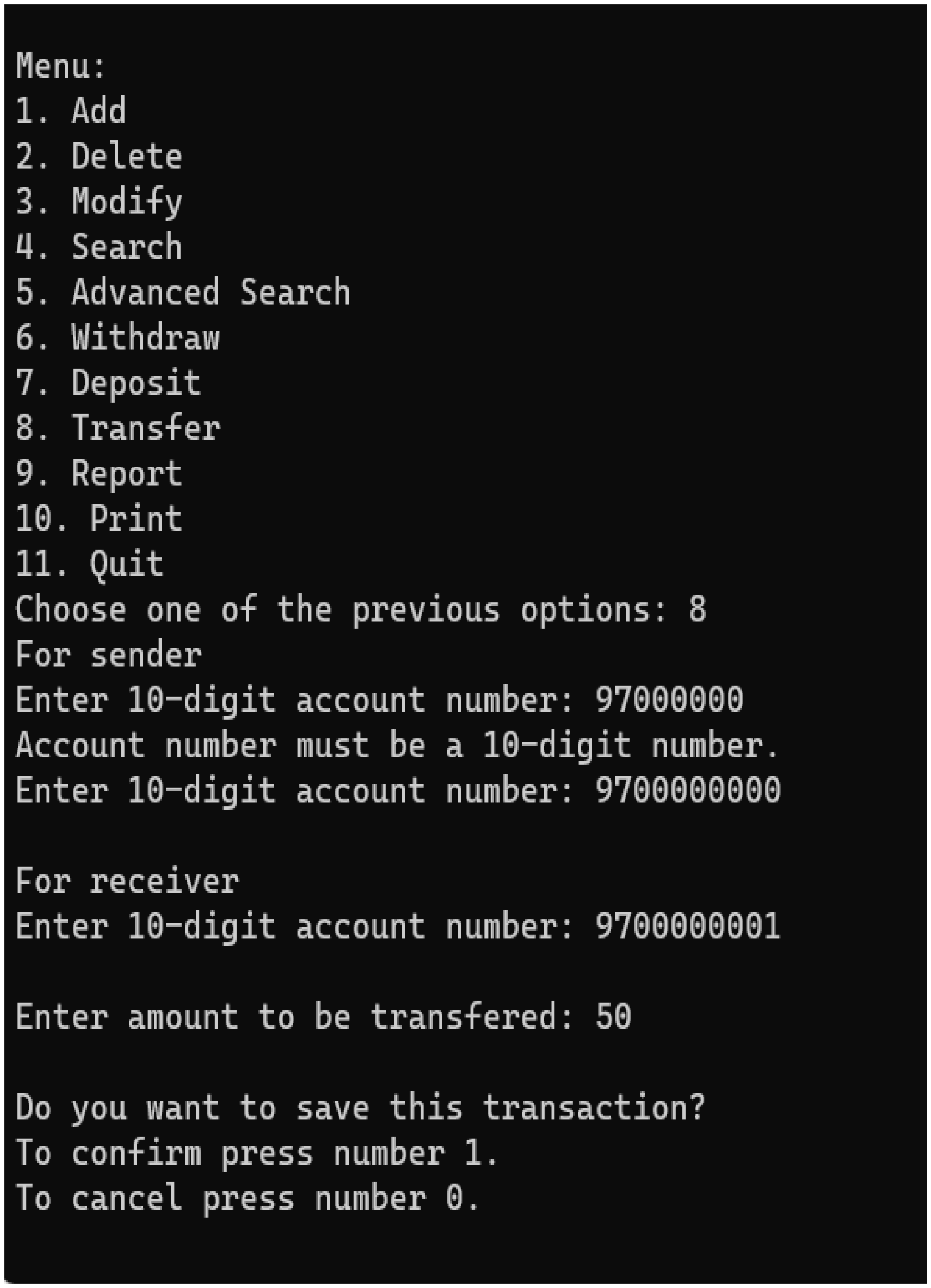


```
Menu:
1. Add
2. Delete
3. Modify
4. Search
5. Advanced Search
6. Withdraw
7. Deposit
8. Transfer
9. Report
10. Print
11. Quit
Choose one of the previous options: 7
Enter 10-digit account number: 9700000000

Maximum deposit limit is 10,000$ per transaction.
Enter amount to be deposited: 524

Do you want to save this transaction?
To confirm press number 1.
To cancel press number 0.
```

8-Transfer

For transfer press 8, you should enter the account number of the sender and then the account number of the receiver

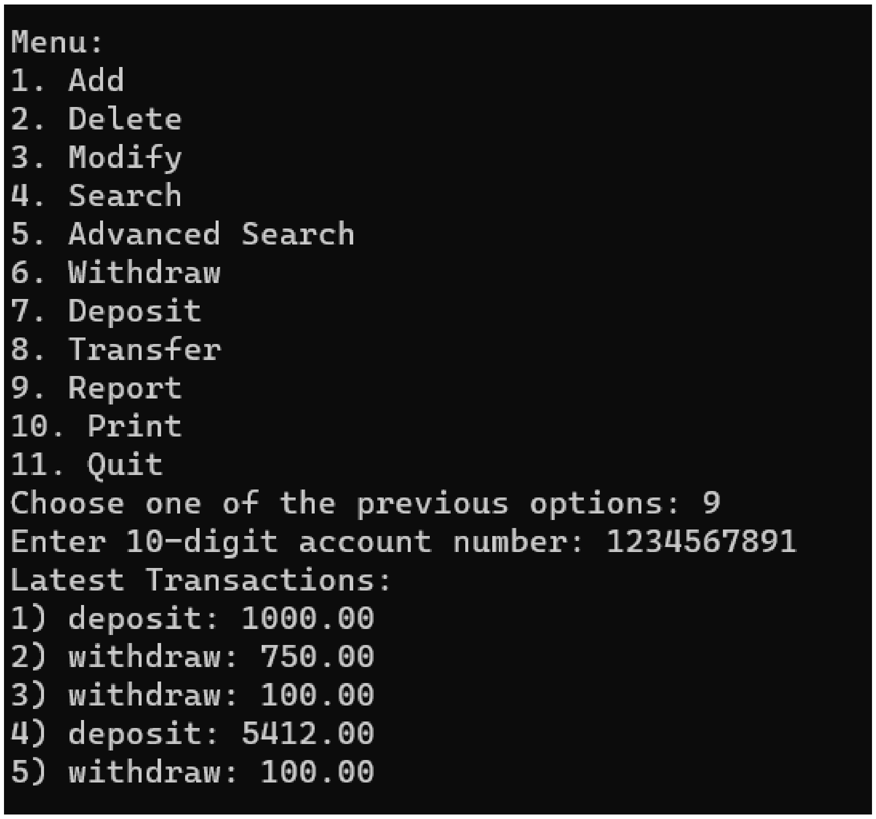
And the amount to be transferred.



```
Menu:
1. Add
2. Delete
3. Modify
4. Search
5. Advanced Search
6. Withdraw
7. Deposit
8. Transfer
9. Report
10. Print
11. Quit
Choose one of the previous options: 8
For sender
Enter 10-digit account number: 97000000
Account number must be a 10-digit number.
Enter 10-digit account number: 9700000000

For receiver
Enter 10-digit account number: 9700000001

Enter amount to be transfered: 50

Do you want to save this transaction?
To confirm press number 1.
To cancel press number 0.
```

9-Report

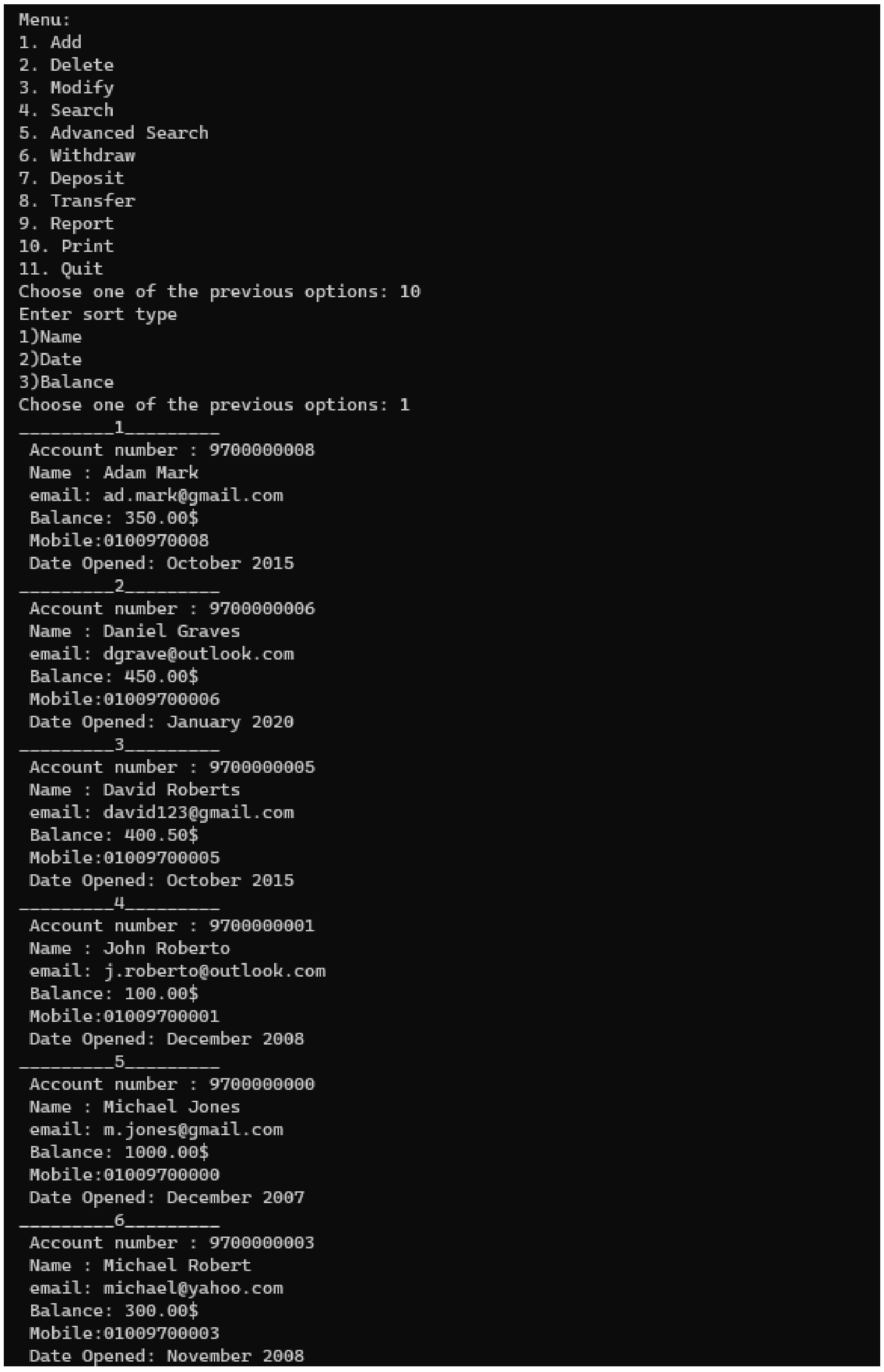
To request a report, press 9. You would be asked to enter the account number you want a report on and it will print the last five transactions this account made.



```
Menu:
1. Add
2. Delete
3. Modify
4. Search
5. Advanced Search
6. Withdraw
7. Deposit
8. Transfer
9. Report
10. Print
11. Quit
Choose one of the previous options: 9
Enter 10-digit account number: 1234567891
Latest Transactions:
1) deposit: 1000.00
2) withdraw: 750.00
3) withdraw: 100.00
4) deposit: 5412.00
5) withdraw: 100.00
```

10-print

Press option 10 then choose the type of sort you want either by name, date or balance.



```
Menu:
1. Add
2. Delete
3. Modify
4. Search
5. Advanced Search
6. Withdraw
7. Deposit
8. Transfer
9. Report
10. Print
11. Quit
Choose one of the previous options: 10
Enter sort type
1)Name
2)Date
3)Balance
Choose one of the previous options: 1
_________1_________
 Account number : 9700000008
 Name : Adam Mark
 email: ad.mark@gmail.com
 Balance: 350.00$
 Mobile:0100970008
 Date Opened: October 2015
_________2_________
 Account number : 9700000006
 Name : Daniel Graves
 email: dgrave@outlook.com
 Balance: 450.00$
 Mobile:01009700006
 Date Opened: January 2020
_________3_________
 Account number : 9700000005
 Name : David Roberts
 email: david123@gmail.com
 Balance: 400.50$
 Mobile:01009700005
 Date Opened: October 2015
_________4_________
 Account number : 9700000001
 Name : John Roberto
 email: j.roberto@outlook.com
 Balance: 100.00$
 Mobile:01009700001
 Date Opened: December 2008
_________5_________
 Account number : 9700000000
 Name : Michael Jones
 email: m.jones@gmail.com
 Balance: 1000.00$
 Mobile:01009700000
 Date Opened: December 2007
_________6_________
 Account number : 9700000003
 Name : Michael Robert
 email: michael@yahoo.com
 Balance: 300.00$
 Mobile:01009700003
 Date Opened: November 2008
```

11-Quit

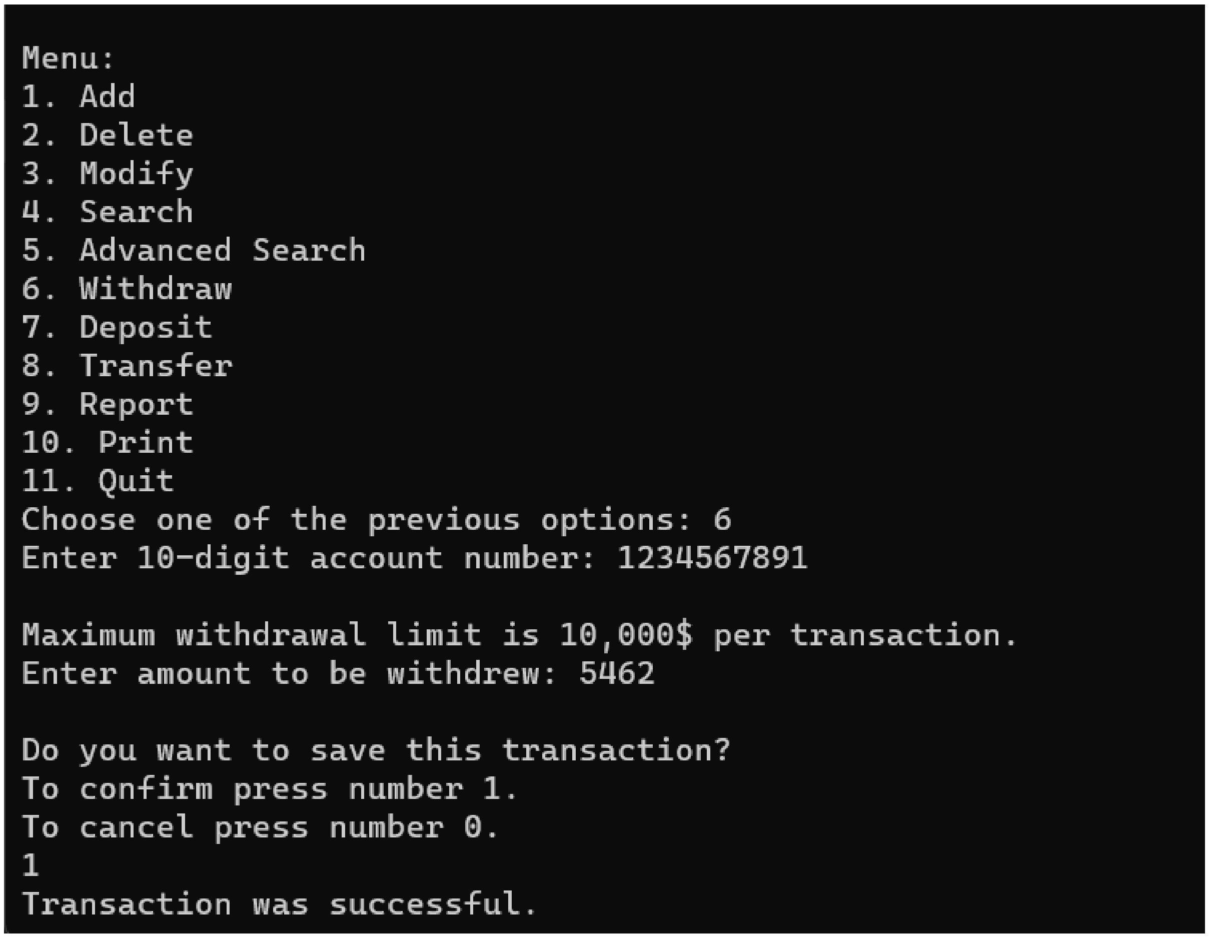
To quit press 11

Otherwise

If you want to go back to the starting menu to login with another account, enter any input other than from 1 to 11.

Saving

After each function done, you will be asked whether you want to save the changes you made or not. Enter the first option to save or the second option to cancel.



```
Menu:
1. Add
2. Delete
3. Modify
4. Search
5. Advanced Search
6. Withdraw
7. Deposit
8. Transfer
9. Report
10. Print
11. Quit
Choose one of the previous options: 6
Enter 10-digit account number: 1234567891

Maximum withdrawal limit is 10,000$ per transaction.
Enter amount to be withdrew: 5462

Do you want to save this transaction?
To confirm press number 1.
To cancel press number 0.
1
Transaction was successful.
```